Two-way automata and regular languages of overlapping tiles

Anne Dicky, David Janin

LaBRI, CNRS UMR 5800
Laboratoire Bordelais de Recherche en Informatique

Rapport de recherche RR-14XX-12

Two-way automata and regular languages of overlapping tiles

July 13, 2012

Anne Dicky, David Janin,
LaBRI, Université de Bordeaux

Abstract

In this paper, we show how the study of two-way automata on words may relevantly be extended to the study of two-way automata on one-dimensional overlapping tiles that generalize finite words. Indeed, over tiles, languages recognizable by finite two-way automata (resp. multi-pebble automata) coincide with languages definable by Kleene's (resp. Kleene's extended) regular expressions.

As an immediate corollary, if we restrict our observations to words, we obtain a new proof of Shepherdson's theorem which posits that every finite state two-way automaton is equivalent to a finite state one-way automaton. We also obtain a new proof that this is still true for two-way automata with pebbles.

Concerning tiles, we show that adding pebbles strictly increases the expressive power of two way automata. The hierarchy induced by the number of allowed pebbles is however shown to collapse to level one. A single pebble is enough to reach maximal expressive power: the class of languages definable in monadic second order logic.

1 Introduction

1.1 Background

In a seminal paper [12], Rabin and Scott defined finite state two-way automata on words. Then Shepherdson proved that the two-way automata have the same expressive power as one-way automata [13]. A few years later, two-way automata were extended with pebbles. Again, these automata have been proved to be no more expressive than one-way automata (see [3] for a state of the art and many more observations).

Despite the negative results, two-way automata have been the subject of many studies. This can be partly explained by their intriguing combinatorial complexity. For instance, the underlying Shepherdson's result has been considered difficult for many years [14]. More precisely, the capacity of two-way automata to read each letter an unbounded number of times makes the structure of two-way automata runs difficult to analyse. This is especially clear in Pécuchet and Birget's algebraic studies of two way-automata [10, 1], in which two-way runs give rise to a rich combinatorial structure. A similar complexity is illustrated by Globerman and Harel's result [3] that the number of allowed pebbles in two-way automata induces a "succintness" hierarchy: each additional pebble provides inherent exponential power.

Still, gaining a full understanding of two-way automata, with or without pebbles, remains a challenging topic, especially if we consider tree-walking automata: the extension of two-way automata to languages of trees. It has recently been shown in a number of studies (see [2] for an overview of these results) that, in the case of trees, each additional pebble provides extra expressive power. Yet, the decidability of this "Pebble" hierarchy is an open problem.

The classical (one-way) finite automata theory have benefited from a rich algebraic language theory that led, and still leads, to many decision algorithms [11]. But there is no algebraic characterization of two-way automata, neither for trees, nor even for finite words [1]. This suggests that the mathematical properties of two-wayness and of (more difficult) pebble-handling have not yet been completely understood even in the simplest case.

1.2 Outline

In this paper, we focus on two-way automata over words, but we define their semantics in terms of *overlapping tiles* [4] instead of words. Tiles that can be seen as domains of partial runs: runs that may start and stop anywhere on the input words. Then we show that the combination of successive partial runs corresponds to a sequential product of tiles, yielding a structure isomorphic to McAlister's inverse monoid [8].

Embedding words into overlapping tiles enables us to apply the most classical techniques used over subsets of the free monoid in a straightforward way. As a result, we show that the regular languages of tiles (definable by Kleene expressions) are exactly the languages of tiles recognizable by finite-state two-way automata. We also prove that the - strictly larger - class of MSO-definable languages of tiles is the class of languages recognizable by finite-state many-pebble automata. Further even, one-pebble automata are shown to capture the whole class. Then, Shepherdson's theorem and analogous results for pebble automata, are obtained as immediate corollaries.

All these results support the long-standing intuition [10, 1, 6] that the theory of inverse monoids [7] *is* a powerful conceptual tool in the study of two-way automata. This is especially illustrated by the fact that all proofs presented here *are* simple.

1.3 Preliminaries

The free monoid. Given a finite alphabet A, A^* denotes the free monoid generated by A, 1 denotes the neutral element. The concatenation of two words u and v is

denoted by uv.

Prefix and suffix lattices. $\leq_p$ stands for the prefix order over A^*, $\leq_s$ for the suffix order. $\vee_p$ (resp. $\vee_s$) denotes the joint operator for the prefix (resp. suffix) order: thus for all words u and v, $u \vee_p v$ (resp. $u \vee_s v$) is the least word whose both u and v are prefixes (resp. suffixes).

The extended monoid $A^* + \{0\}$ (with $0u = u0 = 0$ for every word u), ordered by $\leq_p$ (extended with $u \leq_p 0$ for every word u), is a lattice; in particular, $u \vee_p v = 0$ whenever neither u is a prefix of v, nor v is a prefix of u. Symmetric properties hold in the suffix lattice.

Syntactic inverses. Given $\overline{A}$ a disjoint copy of A, $u \mapsto \overline{u}$ denotes the mapping from $(A + \overline{A})^*$ to itself inductively defined by $\overline{1} = 1$, for every letter $a \in A$, $\overline{a}$ is the copy of a in $\overline{A}$ and $\overline{\overline{a}} = a$, and, for every word $u \in (A + \overline{A})^*$, $\overline{au} = \overline{u}.\overline{a}$. The mapping $u \mapsto \overline{u}$ is involutive ($\overline{\overline{u}} = u$ for every word u); it is an antimorphism of the free monoid $(A + \overline{A})^*$, i.e. for all words u and $v \in (A + \overline{A})^*$, $\overline{uv} = \overline{v}.\overline{u}$.

Free group. The free group $FG(A)$ generated by A is the quotient of $(A + \overline{A})^*$ by the least congruence $\simeq$ such that, for every letter $a \in A$, $a\overline{a} \simeq 1$ and $\overline{a}a \simeq 1$. Let $\preceq$ be the rewriting relation induced by the rules $1 \preceq a\overline{a}$ and $1 \preceq \overline{a}a$ for every $a \in A$. It is well-known that every class $[u] \in FG(A)$ contains a unique element $red(u)$ (the *reduced form* of u) irreducible with respect to $\preceq$, i.e. containing no factor of the form $a.\overline{a}$ or $\overline{a}.a$.

Free inverse monoid. The free inverse monoid $FIM(A)$ generated by A is the quotient of $(A + \overline{A})^*$ by the Wagner congruence $\simeq_W$, i.e. the least congruence such that $u\overline{u}u \simeq_W u$ and $u\overline{u}v\overline{v} \simeq_W v\overline{v}u\overline{u}$ for all $u, v \in (A + \overline{A})^*$.

2 The monoid of overlapping tiles

Here we give a description of the monoid of overlapping tiles. It is shown to be isomorphic to McAlister's monoid [8]. The tight link between (two-way linear) walks on words and tiles is formalized by means of an onto morphism which kernel over walks is proved to be Wagner congruence.

2.1 Positive, negative and context tiles

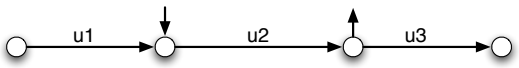
A *tile* over the alphabet A is a triple of words $u = (u_1, u_2, u_3) \in A^* \times (A^* + \overline{A}^*) \times A^*$ such that, if $u_2 \in \overline{A}^*$, its inverse $\overline{u_2}$ is a suffix of u_1 and a prefix of u_3.

When $u_2 \in A^*$ we say that u is a *positive tile*. When $u_2 \in \overline{A}^*$ we say that u is a *negative tile*. When $u_2 = 1$, i.e. when u is both positive and negative, we say that

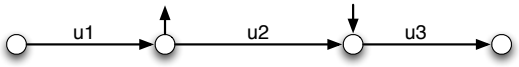
u is a *context tile*. Sets T_A, T_A^+, T_A^- and C_A will respectively denote the set of tiles, the set of positive tiles, the set of negative tiles and the set of context tiles over A.

The *domain* of a tile $u = (u_1, u_2, u_3)$ is the reduced form of $u_1u_2u_3$ (always a word of A^*); its *root* is the word u_2.

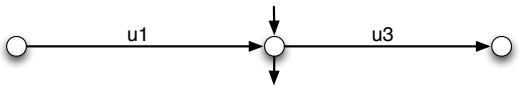
A positive tile $u = (u_1, u_2, u_3)$ is conveniently drawn as a (linear, unidirectional and left to right) Munn's birooted word tree [9]:



where the dangling input arrow (marking the beginning of the root) appears on the left of the dangling output arrow (marking the end of the root). A negative tile of the form $u = (u_1u_2, \overline{u_2}, u_2u_3) \in A^* \times \overline{A}^* \times A^*$ is also drawn as a birooted word tree



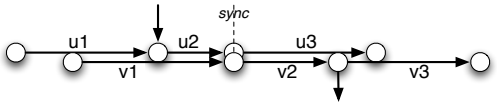
where the dangling input arrow appears on the right of the dangling output arrow. A context tile of the form $u = (u_1, 1, u_3) \in A^* \times 1 \times A^*$ is then drawn as follows:



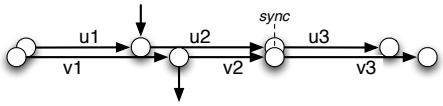
2.2 The inverse monoid of tiles

In this part, we abusively denote a word u of $A^* + \overline{A}^*$ by any word of $(A + \overline{A})^*$ whose reduced form is u.

Intuitively, the sequential product of two tiles is their superposition in such a way that the end of the root of the first tile coincides with the beginning of the root of the second tile; the superposition requires pattern-matching conditions to the left and to the right of the synchronization point. When both tiles are positive, this can be drawn as follows:



The product can be extended to arbitrary tiles, as illustrated by the following figure (positive u and negative v):



Formally, we extend the set T_A with a zero tile to obtain $T_A^0 = T_A + \{0\}$. The sequential product of two non-zero tiles $u = (u_1, u_2, u_3)$ and $v = (v_1, v_2, v_3)$ is defined as

$$u.v = ((u_1u_2 \vee_s v_1)\overline{u_2}, u_2v_2, \overline{v_2}(u_3 \vee_p v_2v_3))$$

when both $u_1u_2 \vee_s v_1 \neq 0$ and $u_3 \vee_p v_2v_3 \neq 0$, and $u.v = 0$ otherwise. We let $u.0 = 0.u = 0$ for every $u \in T_A^0$.

Remark. Let a, b, c and $d \in A$ be distinct letters. Then $(a, b, c).(b, c, d) = (a, bc, d)$ whereas $(a, b, c).(a, c, d) = 0$. In the latter case, the left matching constraint is violated because $a \neq b$.

Theorem 1 *Set T_A^0 equipped with the sequential product of two tiles is an inverse monoid with neutral element $1 = (1, 1, 1)$ and (pseudo) inverses given by $0^{-1} = 0$ and for every non zero tile $u = (u_1, u_2, u_3) \in T_A$, $u^{-1} = (u_1u_2, \overline{u}_2, u_2u_3)$.*

Proof. Since 1 is obviously the neutral element and 0 the absorbant element, we have to prove that the sequential product is well-defined (or sound) and associative.

Soundness. We first prove that the sequential product of two tiles is a tile. Let $u = (u_1, u_2, u_3)$ and $v = (v_1, v_2, v_3)$ be tiles such that $u.v \neq 0$. In all case, with u or v positive or negative, one can check that $(u_1u_2 \vee_s v_1).\overline{u}_2 \in A^*$, $\overline{v}_2(u_3 \vee_s v_2v_3) \in A^*$. Moreover, when $u_2v_2 \in \overline{A}^*$ (with elements of the free group $FG(A)$ always reduced), we also have $\overline{u_2v_2} = \overline{v_2}.\overline{u_2} \leq_s (u_1u_2 \vee_s v_1).\overline{u}_2$ and $\overline{u_2v_2} = \overline{v_2}.\overline{u_2} \leq_p \overline{v}_2(u_3 \vee_s v_2v_3)$. We conclude thus that $u.v = ((u_1u_2 \vee_s v_1)\overline{u_2}, u_2v_2, \overline{v_2}(u_3 \vee_p v_2v_3))$ is indeed a tile.

Associativity. We observe first that the pattern-matching conditions are associative: if $u = (u_1, u_2, u_3)$, $v = (v_1, v_2, v_3)$ and $w = (w_1, w_2, w_3)$ are non-zero tiles, the products $(u.v).w$ and $u.(v.w)$ are non-zero tiles if and only if $u_1u_2v_2 \vee_s v_1v_2 \vee_s w_1 \neq 0$ and $u_3 \vee_s v_2v_3 \vee_s v_2w_2w_3 \neq 0$. In that case, $(u.v).w = ((u_1u_2v_2 \vee_s v_1v_2 \vee_s w_1)\overline{v_2}\,\overline{u_2}, u_2v_2w_2, \overline{w_2}\,\overline{v_2}(u_3 \vee_s v_2v_3 \vee_s v_2w_2w_3))$ which just equals $u.(v.w)$.

To prove that T_A^0 is an inverse monoid (proving that every element as a unique pseudo inverse), it suffices to prove [7] that every element of T_A has a (pseudo) inverse and that idempotents commutes.

Since 0 is obviously a pseudo inverse to itself and commutes with every elements we only consider the case of non zero tiles.

Existence of pseudo inverses. For every $u = (u_1, u_2, u_3) \in T_A$, given $u^{-1} = (u_1u_2, \overline{u}_2, u_2u_3)$, we do have $u.u^{-1}.u = u$ and $u^{-1}.u.u^{-1} = u$. Indeed, by symmetry, since $(u^{-1})^{-1} = u$ it suffices to prove that $u.u^{-1}.u = u$. But one can easily check

that $u.u^{-1} = (u_1, 1, u_2u_3)$ (equivalently $u^{-1}u = (u_1u_2, 1, u_3)$) and thus $uu^{-1}u = (u_1, u_2, u_3)$.

Commutation of idempotents. Since context tiles C_A form a commutative submonoid of T_A (this immediately follows from the commutativity of $\vee_s$ and $\vee_p$) it suffices to prove that the idempotents in T_A are the context tiles.

Let $u \in C_A$. By definition $u = (u_1, 1, u_3)$ hence $u.u = ((u_1.1 \vee_s u_1).\overline{1}, 1.1, \overline{1}.(u_3 \vee_p 1.u_3))$ and thus $u.u = u$. Conversely, let $u = (u_1, u_2, u_3) \in T_A$. Assume that $u.u = u$. By product definition, this means in particular $u_2u_2 = u_2$ hence $u_2 = 1$ and thus $u \in C_A$. □

An immediate property worth being mentioned:

Lemma 2 *The mapping $u \mapsto (1, u, 1)$ from A^* to T_A is a one-to-one morphism.*

In other words, the free monoid A^* can be seen as a submonoid of T_A^0. In the remainder of the text we may use the same notation for words of A^* and their images in T_A^0.

Remark. In [8], Lawson already provides a description of overlapping tiles. Non zero tiles are modeled as triples of the form $(u_1, u_2, u_3) \in A^* \times A^* \times A^*$ with $u_1 \leq_p u_2$ and $u_3 \leq_s u_2$. One can show that the mapping $(u_1, u_2, u_3) \mapsto (u_1, red(\overline{u_1}.u_2.\overline{u_3}), u_3)$ that maps Lawson's models to the model presented here is, extended to 0, a monoid isomorphism. In other words, from a mathematical point of view, this hardly makes any differences. However, one may think, as the present authors do, that the model proposed in this paper induces a simpler product definition.

2.3 Linear walks and the McAslister monoid

We provide here a proof that T_A is isomorphic to McAlister monoid. Again, this follows from Lawson's characterization of McAlister monoid [8] that can be shown isomorphic to ours. However, the proof given here, via the monoid W_A^0 of linear walks, conveys most of the intuition about the link with two-way automata.

Informally, a *walk* over a word $u \in A^*$ is a word of $(A + \overline{A})^*$ corresponding to a back and forth reading of u (left to right reading is modeled by letters of A, and right to left reading by letters of $\overline{A}$). Not all words of $(A+\overline{A})^*$ are walks. Obviously, no factor $a\overline{b}$ or $\overline{a}b$ with distinct letters a and b can occur in a walk. But things are a little more complex. A word like $ba\overline{a}\overline{c}$ with distinct a, b and c is still not a walk. It occurs that walks are easily defined by contraposition.

Formally, let $\perp$ be the set of all words $v \in (A+\overline{A})^*$ such that there exists a word $u \preceq v$ and some distinct letters a and b such that either $a\overline{b}$ or $\overline{a}b$ is a factor of u.

A (linear) walk is any word $u \in (A + \overline{A})^*$ such that $u \notin \perp$. Clearly, $\perp$ is closed by product with arbitrary elements of $(A + \overline{A})^*$. It is thus an ideal of $(A + \overline{A})^*$.

We define the *monoid of walks* W_A^0 as the Rees quotient $(A + \overline{A})^*/\perp$. It is the monoid obtained from $(A + \overline{A})^*$ by merging all elements of $\perp$ into a zero, the undefined walk. The set of non zero walks is denoted by W_A. By definition, $W_A = (A + \overline{A})^* - \perp$. It shall be clear that $\perp$ (henceforth W_A) is closed under Wagner congruence.

Walks and tiles are related by the following Lemma:

Lemma 3 *For every non zero walk $w \in W_A$, there is a unique tile $\theta(w) = (u_1, u_2, u_3) \in T_A$ such that $\overline{u_1}u_1u_2u_3\overline{u_3} \simeq_W w$. In particular, for every walks w_1 and w_2, $w_1 \simeq_W w_2$ if and only if $\theta(w_1) = \theta(w_2)$.*

Moreover, extending θ to a mapping from W_A^0 to T_A^0 by taking $\theta(\perp) = 0$, θ is an onto monoid morphism, i.e. in particular, for every defined walks w and $w' \in W_A$ such that $w.w' \in W_A$, $\theta(w).\theta(w') = \theta(w.w')$.

Proof. Existence. Let $w \in W_A$. The existence of some $\theta(w)$ as above is proved by induction over the number $n(w)$ of turns (or alternation of positive and negative letters) in w. Observe that as soon as we prove the existence of $\theta(w)$ we can take $\theta(\overline{w}) = (\theta(w))^{-1}$. It follows that we only need to prove the existence of either some $\theta(w)$ or some $\theta(\overline{w})$.

When $n(w) = 0$, if $w \in A^*$ we take $\theta(w) = (1, w, 1)$. By symmetry, if $w \in \overline{A}^*$ we take $\theta(w) = (w, \overline{w}, w)$.

Assume $n(w) > 0$. We have $w = xw'$ for some $x \in A^* + \overline{A}^*$ and $w' \in W_A$ and $n(w') = n(w)-1$. By induction hypothesis, $\theta(w')$ exists with $\theta(w') = (u_1, u_2, u_3)$ and $w' \simeq_W \overline{u}_1u_1u_2u_3\overline{u}_3$ and thus, since Wagner is a congruence, $w \simeq x\overline{u}_1u_1u_2u_3\overline{u}_3$. Now, since W_A is closed under Wagner congruence, this means that $x\overline{u}_1u_1u_2u_3\overline{u}_3 \in W_A$ hence, in particular, $x\overline{u}_1 \in W_A$.

By symmetry, possibly taking $\overline{w}$ instead of w, we may assume that $x \in A^*$. Since $x\overline{u}_1 \in W_A$ this means that $x \vee_s u_1 \neq 0$. It follows that $(1, x, 1).\theta(w') = ((x \vee_s u_1)\overline{x}, xu_2, u_3) \neq 0$ and, in both cases $u_1 \leq_s x$ or $x \leq_s u_1$, we take $\theta(w) = (1, x, 1).\theta(w')$.

Indeed, in the first case, $u_1 = u_1'x$ for some $u_1' \in A^*$. We have $(1, x, 1).\theta(w') = (u_1', xu_2, u_3)$ and we know that $w \simeq_W x\overline{(u_1'x)}(u_1'x)u_2u_3\overline{u}_3$ hence $w \simeq_W x\overline{x}\overline{u}_1'u_1'xu_2u_3\overline{u}_3$ hence $w \simeq_W \overline{u}_1'u_1'x\overline{x}xu_2u_3\overline{u}_3$ hence $w \simeq_W \overline{u}_1'u_1'xu_2u_3)$.

In the second case, $x = x'u_1$ for some $x' \in A^*$. We have $(1, x, 1).\theta(w') = (1, xu_2, u_3)$ and $w \simeq_W (x'u_1)\overline{u}_1u_1u_2u_3\overline{u}_3$ hence $w \simeq_W x'u_1u_2u_3\overline{u}_3$ hence $w \simeq_W xu_2u_3\overline{u}_3$.

The fact that, in both case, $\theta(w)$ is a well-defined tiles (which can also be proved following a tedious case studies) just follows from the soundness of our tile product.

Unicity. Let $w \in W_A$ and two decomposition $w \simeq_W u = \overline{u}_1 u_1 u_2 u_3 \overline{u}_3$ and $w \simeq_W u' = \overline{u}'_1 u'_1 u'_2 u'_3 \overline{u}'_3$. Since Wagner congruence is compatible with the free group reduction this implied that $red(u) = red(u')$ hence $u_2 = u'_2$.

Let then a be an arbitrary letter in A. On the left side of w, since $au_1 u \in W_A$, we also have $au_1 u' \in W_A$. Hence $au_1 \vee_s u'_1 \neq 0$. But since this holds for arbitrary $a \in A$ (assumed to have two distinct letters) this implies that $u'_1 \leq_s u_1$. By symmetry we also have $u_1 \leq_s u'_1$ hence $u_1 = u'_1$. An analogous argument holds for the right side proving thus that $u_3 = u'_3$.

Compositionality. The fact that θ extended to 0 is a monoid morphism immediately follows from the proof of its existence. Moreover, for every non zero tile $(u_1, u_2, u_3) \in T_A$ one has $\theta(\overline{u}_1 u_1 u_2 u_3 \overline{u}_3) = (u_1, u_2, u_3)$ hence θ is onto. □

Remark. In some sense, this Lemma captures most of the combinatorial analysis of two-way automata runs made in [10, 1]. However, being only concerned by *what* do read two-way automata rather than *how* do they perform readings, Wagner equivalence appears as the appropriate simplification concept.

Previous lemma says that, in W_A, Wagner congruence is the kernel of θ. Now, since $\perp$ is closed under the Wagner congruence, it can also be seen as an ideal of the free inverse monoid $FIM(A)$ and then $FIM(A)/\perp$ just corresponds to McAlister original definition of his monoid.

Corollary 4 *The monoid of tiles T_A^0, isomorphic to the quotient of walks $W_A^0/\simeq_W$ by Wagner congruence, is also isomorphic to McAlister monoid $FIM(A)/\perp$.*

The relationship between walks, tiles and elements of the free inverse monoid is summarized by the following commuting diagram.

$$\begin{array}{ccc} (A+\overline{A})^* & \overset{[\ ]_{\simeq_W}}{\twoheadrightarrow} & FIM(A) \\ \Big\downarrow{\scriptstyle /\perp} & & \Big\downarrow{\scriptstyle /\perp} \\ W_A^0 & \xrightarrow{\quad\theta\quad} & T_A^0 \end{array}$$

3 Classes of definable languages of tiles

We define here various notions of regular languages of tiles (or walks). Following a classical habit in formal language theory, for any s some monoid M, we denote by s, depending on the context of use, both the element s or the singleton $\{s\}$.

3.1 Regular and k-regular languages

As for languages of words, the following operators are defined over languages of tiles. For all $M, N \subseteq T_A$, let

- Sum: $M + N = \{u \in T_A : u \in M \vee u \in N\}$
- Product: $M.N = \{u.v \in T_A : u \in M, v \in N\}$
- Star: $M^* = \sum_{n\in\mathbb{N}} M^n$ with $M^0 = \{(1,1,1)\}$ and $M^{k+1} = M.M^k$ for every $k \in \mathbb{N}$.

On purpose, we restrict to non zero tiles. Still one can check that most classical properties are satisfied. Sum and product of languages are associative and product distributes over sum, i.e. for every language L, M and $N \subseteq T_A$, $L.(M + N) = L.M + L.N$ and $(M + N).L = M.L + N.L$. Moreover, for all languages M and $N \subseteq T_A$, $M^*.N$ is the least solution with respect to inclusion of the language equation $X = M.X + L$.

Definition. A language of tiles $M \subseteq T_A$ is *regular* when it can be defined as the result of finitely many additions, multiplications and star operations of finite languages of non-zero tiles. The class of regular languages of tiles is denoted by *REG*.

Are there other operators on languages worth being defined ? An obvious one is the inverse operator. For every $M \subseteq T_A$ let $M^{-1} = \{u^{-1} \in T_A : u \in M\}$. However, the fallowing fact is well-known in the theory of inverse monoids:

Lemma 5 *The class REG of regular tile languages is closed under the inverse operation.*

Proof. Just observe that for all $L, M \subseteq T_A$, we have $(L + M)^{-1} = L^{-1} + M^{-1}$, $(L.M)^{-1} = M^{-1}.L^{-1}$ and $(L^*)^{-1} = (L^{-1})^*$. □

This suggests that we need more. Classical and inverse operators on languages are completed by the following unary operator. For every $M \subseteq T_A$, the *context projection* M^C of M is defined by $M^C = M \cap C_A$.

Definition. For every $k \in \mathbb{N}$, a language $M \subseteq T_A$ is called k-regular when either $k = 0$ and M is regular or $k = k' + 1$ and M can be defined as the result of finitely many additions, multiplications, star and *inverse* operations of k'-regular languages N and context projection N^C of k'-regular languages. The class of k-regular languages is denoted by k-REG.

As an immediate consequence of the definition:

Corollary 6 *REG $\subseteq$ 1-REG $\subseteq$ 2-REG $\subseteq$ 3-REG $\subseteq$ k-REG $\subseteq \cdots$*

3.2 Languages definable in MSO

Last, the bi-rooted presentation of every non zero tile can also be seen as a FO-structure. It follows that one also define the class MSO of languages definable by means of Monadic Second Order formula. The following two Theorems are proved in [4].

Theorem 7 (Robustness [4]) *For all M and $N \subseteq T_A$, if both M and N are MSO-definable then so are $M + N$, $M.N$, M^*, M^{-1} and M^C.*

Corollary 8 *For every $k \in \mathbb{N}$, k-REG $\subseteq$ MSO.*

Theorem 9 (Simplicity [4]) *A language $M \subseteq T_A$ is MSO-definable if and only there are finitely many triples of regular languages of words $(L_i, C_i, R_i)_{i \in I \cup J} \subseteq \mathcal{P}(A^*) \times \mathcal{P}(A^*) \times \mathbb{P}(A^*)$ such that $M = \Sigma_{i \in I}(L_i \times C_i \times R_i)^{e_i}$ with either $e_i = 1$ or $e_i = -1$.*

Corollary 10 *The class 1-REG and MSO are equal.*

Proof. For every triples of languages of words L, C and $R \subseteq A^*$, embedding them in T_A, one has $(L \times C \times R) = (L^{-1}.L)^C.C.(R.R^{-1})^C$. It follows that, provided L, C and R are regular, their embeddings in T_A are also regular (or 0-regular) and thus both $(L \times C \times R)$ and $(L \times C \times R)^{-1}$ are 1-regular. Then, by Theorem 9, this proves that MSO $\subseteq$ 1-REG. □

Remark. To complete the picture, the class REC of languages of tiles recognizable by means of finite monoid can also be defined. This class is studied in [4]. It is shown to be strongly related with bi-infinite periodic words. As a consequence, it can be shown that REC is strictly included REG.

4 Two-way automata and regular languages

We prove here that regular languages of tiles are just the languages recognizable by finite two-way automata.

4.1 Two-way automata

A finite-state two-way automaton (or 2WA for short) on an alphabet A is a quadruple $\mathcal{A} = \langle Q, I, F, \Delta \rangle$ with a finite set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, and a transition table $\Delta : (A + \overline{A}) \to \mathcal{P}(Q \times Q)$.

A *run* of $\mathcal{A}$ over a string (of $(A + \overline{A})^*$ is a finite sequence $\rho = q_0 a_1 q_1 \dots q_{n-1} a_n q_n$ where $n \geq 0$, $q_0, \dots, q_n \in Q$ and $a_1, \dots, a_n \in A + \overline{A}$, such that for every $1 \leq i \leq n$, $(q_{i-1}, q_i) \in \Delta(a_i)$.

The (string) run ρ is *accepting* if $q_0 \in I$ and $q_n \in F$. In that case, we say that the associated string $s_\rho = a_1 \cdots a_n \in A^*$ is accepted by the automaton $\mathcal{A}$. The string language of strings accepted by $\mathcal{A}$ is written $S(\mathcal{A})$. In that case, automaton $\mathcal{A}$ is just seen as a classical one-way automaton on the (uninterpreted) alphabet $A + \overline{A}$.

We say that run ρ is a run over a walk when, moreover, s_ρ is a walk. In that case, we write w_ρ for the string s_ρ. The set of walks accepted by $\mathcal{A}$ is written $W(\mathcal{A})$. In other words: $W(\mathcal{A}) = S(\mathcal{A}) \cap W_A$.

Remark. Two-way automata may be defined with the possibility to stand still at the same position while changing state. Clearly, these "silent" transitions are just syntactic sugar that can be replaced with extra non-silent transitions (by the same techniques used to eliminate classical silent transitions in one-way automata). Our definition follows [3].

The following lemma emphasizes the difference between two-way automata (interpreted on walks) and ordinary finite-state automata (interpreted on strings):

Lemma 11 *There exists a two-way automaton $\mathcal{A}$ such that $W(\mathcal{A})$ is not a regular language.*

Proof. Assume $A = \{a, b\}$ with two distinct letters, let $Q = \{q_0, q_1, q_2, q_3\}$ be a four-state set, and let $\mathcal{A} = \{Q, \{q_0\}, \{q_3\}, \Delta\}$ with $\Delta(a) = \{(q_0, q_1)\}$, $\Delta(b) = \{(q_1, q_1)\}$, $\Delta(\overline{b}) = \{(q_1, q_2), (q_2, q_2)\}$ and $\Delta(\overline{a}) = \{q_2, q_3\}$.

$S(\mathcal{A}) = ab^+\overline{b}^+\overline{a}$. It follows that $W(\mathcal{A}) = \{ab^n\overline{b}^n\overline{a} : n > 0\}$, hence $W(\mathcal{A})$ is non regular. □

4.2 Word and tile languages recognized by 2WA

The *language of words* $L(\mathcal{A})$ recognized by a two-way automaton $\mathcal{A}$ on alphabet A (completed by $\overline{A}$) is the set of *words* $u \in A^*$ such that there is an accepting run of $\mathcal{A}$ corresponding to a back-and-forth reading of u, i.e. a walk $w \in \mathcal{A}$ such that $w \simeq_W u$. In other words: $L(\mathcal{A}) = \{u \in A^* : (1, u, 1) \in T(\mathcal{A})\}$.

In a similar way, we define the *language of tiles* $T(\mathcal{A})$ recognized by a two-way automaton $\mathcal{A}$ as the set of tiles $u = (u_1, u_2, u_3) \in T_A$ such that there exists a walk $w \in W(\mathcal{A})$ such that $w \simeq_W \overline{u_1}u_1u_2u_3\overline{u_3}$, i.e. w is a walk over $u_1u_2u_3$, starting at the entry point of the tile u and ending at its exit point. In other words: $T(\mathcal{A}) = \theta(W(\mathcal{A}))$.

We now prove a Kleene theorem for tile languages.

Theorem 12 *The regular tile languages are exactly the tile languages recognizable by finite-state two-way automata.*

Proof. This follows from Lemmas 15 and 13 below. □

Lemma 13 *Every regular language of tiles $M \subseteq T_A$ is recognizable by a finite state two-way automaton on the alphabet A.*

Proof. For any finite tile language $M \subseteq T_A$, any one-way automaton recognizing the finite word language $\{\overline{u_1}u_1u_2u_3\overline{u_3} : (u_1, u_2, u_3) \in M\}$ can be viewed as a two-way automaton recognizing M. Thus all finite languages of non-zero tiles are recognizable.

More generally, any finite-state one-way automaton $\mathcal{A}$ over the alphabet $A + \overline{A}$, recognizing a word language $L \subseteq (A + \overline{A})^*$, can be viewed as a 2WA recognizing the tile language $\theta(L \cap W_A)$.

For all word languages $L, M \subseteq (A + \overline{A})^*$ we have

- $\theta((L + M) \cap W_A) = \theta(L \cap W_A) + \theta(M \cap W_A)$ (obvious)
- $\theta(LM \cap W_A) = \theta(L \cap W_A).\theta(M \cap W_A)$ (from Lemma 3)
- $\theta(L^* \cap W_A) = (\theta(L \cap W_A))^*$ (consequence of definition and both cases above).

We conclude the proof by induction on the structure of regular expressions, using the classical one-way automata theory. □

Let $\mathcal{A} = \langle Q, I, F, \Delta\rangle$ be a two-way automaton on the alphabet A. For every pair of states $(p, q) \in Q \times Q$, let $T_{p,q}$ denote the language of tiles recognized by the two-way automaton $\mathcal{A}_{p,q} = \langle Q, \{q\}, \{p\}, \Delta\rangle$.

Lemma 14 *The sets $T_{p,q}$ with p and $q \in Q$ are the least solution (w.r.t. inclusion order) of the system of equations*

$$T_{p,q} = \delta_{p,q} + \sum_{a \in A+\overline{A}} \sum_{(p,r) \in \Delta(a)} \theta(a).T_{r,q} \qquad (E_{p,q})$$

where $\delta_{p,q} = \{1\}$ if $p = q$ and $\emptyset$ otherwise.

Proof. For every state p and q, and every integer $k \geq 0$, let $W^k_{p,q}$ denote the set of walks of length at most k accepted by $\mathcal{A}_{p,q}$. The identities $W^0_{p,q} = \delta_{p,q}$ and

$$W^{k+1}_{p,q} = W^k_{p,q} + \sum_{a \in A+\overline{A}} \sum_{(p,r) \in \Delta(a)} (a.W^k_{r,q} \cap W_A)$$

immediately follow from the definition of the walk acceptance. Then, with $W_{p,q} = \sum_{k \in \mathbb{N}} W^k_{p,q}$, by Tarski's fixpoint theorem, the sets $W_{p,q}$ form the least fixed point of the system of equations:

$$W_{p,q} = \delta_{p,q} + \sum_{a \in A+\overline{A}} \sum_{(p,r) \in \Delta(a)} (a.W_{r,q} \cap W_A) \qquad (E'_{p,q})$$

We conclude the proof by taking the images by θ of all equations. □

Lemma 15 *$T(\mathcal{A})$ is a regular tile language.*

Proof. The least solution of the system of equations $E_{p,q}$ can be computed by a Gaussian elimination of variables, applying the fact that the least solution of an equation of the form $X = U.X + V$ in $\mathcal{P}(T_A)$ is $U^*.V$. This gives a regular expression of every $T_{p,q}$ and thus for $T(\mathcal{A})$ as well since $T(\mathcal{A}) = \Sigma_{(p,q) \in I \times F} T_{p,q}$. □

Corollary 16 (Shepherdson) *Every language of words recognizable by a two-way automata is regular.*

Proof. Let $\mathcal{A}$ be a finite two-way automata. The language $L(\mathcal{A}) \subseteq A^*$ of words recognized by $\mathcal{A}$ is defined to be $L(\mathcal{A}) = \{u \in A^* : (1, u, 1) \in T(\mathcal{A}) : u_1 = u_3 = 1\}$. Now let $\#$ be a new letter. By Lemma 15, language $\#.T(\mathcal{A}).\#$ is regular. Moreover, since $\# \notin A$ we also have $\#.T(\mathcal{A}).\# \subseteq \{1\} \times \#A^*\# \times \{1\}$ and thus $\#.T(\mathcal{A}).\# = \#.L(\mathcal{A}).\#$. Then we conclude by applying Theorem 9. □

5 Pebble automata and k-regular languages

In this section, we prove that k-regular languages are captured by pebble automata. The number of pebbles is not explicitly encoded into our definition of pebble automata (like invisible pebbles in [2]). It follows that k-pebble automata are (equivalently) defined, in our approach, as (invisible) pebble automata restricted to runs using at most k pebbles.

5.1 Multi-pebble automata

A finite-state pebble automaton (or PWA for short) on an alphabet A is a quadruple $\mathcal{A} = \langle Q, I, F, \Delta \rangle$ with a finite set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, and a transition table $\Delta : (A + \overline{A} + \{1^+, 1^-\}) \to \mathcal{P}(Q \times Q)$. For every $a \in A + \overline{A}$, $\Delta(a)$ tells how a can be read as in a two-way automaton. Newly, $\Delta(1^+)$ tells how a pebble can be left on the current position and $\Delta(1^-)$ tells how a pebble can be removed. In other words, a pebble automaton is a two-way automaton that has the capacity, from time to time, to leave and remove pebbles placed *between* letters of the underlying word.

Of course, a pebble cannot be removed if it has not been left before. Moreover, the behaviour of pebble automata must be restricted so that only the last pebble put on a word may be removed (in a LIFO style). Otherwise, pebble automata can define languages of accepting runs of Turing machines.

These two restrictions are conveniently captured by the following definition of pebble automata runs.

A *position configuration* is a non-empty finite sequence of (positive or negative) integers $p = n_0.\cdots.n_k \in \mathbb{Z}^+$. The intended meaning of position configuration p is that n_i records the relative number of letters (positive or negative) read from the ith pebble left on the input word, with the initial starting point modeled as a sort of a 0th pebble.

Moreover, since any non-zero pebble is eventually removed in a run, we only record the position relative to the last pebble left. Pushing a pebble on the stack freezes the previous recorded relative positions. It follows that, at any time, $n_i + n_{i+1} + \cdots + n_k$ will denote the number of positive (or negative) letters that separate the automaton head from the position of the ith pebble. In particular, when $n_k = 0$, the kth pebble can be removed.

Formally, a *run* of the pebble automaton $\mathcal{A}$ is a finite word $\rho \in ((Q \times \mathbb{Z}^+).(A + \overline{A} + 1))^*.(Q \times \mathbb{Z}^+)$ such that, for every factor of ρ of the form $(q, p).a.(q', p')$, with

$a \in A + \overline{A} + 1$, one of the following conditions is satisfied:

- $(q, q') \in \Delta(a)$, $a \in A + \overline{A}$ and $p' = p.\delta_a$, with $\delta_a = 1$ if $a \in A$ and $\delta_a = -1$ if $a \in \overline{A}$,
- $(q, q') \in \Delta(1^+)$, $a = 1$ and $p' = p.0$,
- $(q, q') \in \Delta(1^-)$, $a = 1$ and $p = p'.0$,

As before, we also assume that the projection w_ρ of ρ to $(A+\overline{A})^*$ is a non-zero walk.

We observe that the position configurations are handled as a (left to right) stack. Leaving a pebble amounts to pushing 0, the new relative position of the head from that pebble. Reading $a \in A + \overline{A}$, the relative position from the last left pebble is changed by δ_a. Removing a pebble amounts to popping the last relative position of the head from that pebble. This relative position is forced to 0. This way, we model the fact that the head must have moved back on the position where the pebble has been left.

The number of pebbles used in a run $\rho \in ((Q\times\mathbb{Z}^+).(A+\overline{A}))^*.(Q\times\mathbb{Z}^+)$ is defined to be the least integer $k \in \mathbb{N}$ such that $\rho \in ((Q \times \mathbb{Z}^{\leq k+1}).(A + \overline{A}))^*.(Q \times \mathbb{Z}^{\leq k+1})$ where $\mathbb{Z}^{\leq k}$ stands for the non sequences of integers of length at most k.

Still writing w_q for the projection of run ρ on the alphabet $A + \overline{A}$, we say that a triple $u = (u_1, u_2, u_3)$ is accepted by automaton $\mathcal{A}$ with at most k-pebble when there exists run ρ using at most k pebbles such that $w_\rho \simeq_W \overline{u_1}u_1u_2u_3\overline{u_3}$ with start state of the form $(q, 0)$ with $q \in I$ and end state of the form (q', i) with $q' \in F$. A simple check of our definition shows that, in that case $i = |u_2|$ when $u_2 \in A^*$ and $i = -|u_2|$ when $u_2 \in \overline{A}^*$.

5.2 Pebbles vs tiles context operators

From now on, a k-pebbles automaton is defined as a many-pebble automaton which runs are only allowed to use at most k pebbles.

Theorem 17 *For every $k \in \mathbb{N}$, the k-regular tile languages are exactly the tile languages recognizable by finite-state k-pebbles automata.*

Proof. Follows from the Lemmas 18 and 19 bellow. □

Lemma 18 *Every language of tiles definable by a finite k-pebbles automaton is k-regular.*

Proof. Let $\mathcal{A} = \langle Q, I, F, \delta \rangle$ be a finite many-pebble automaton. For every pair of states $(p, q) \in Q \times Q$, every integer $k \geq 0$, let $T^k_{p,q} \subseteq T_A$ be the language of tiles recognized by automaton $\mathcal{A}$ with at most k-pebbles from state p to state q. Let also $C^k_{p,q}$ be the associated set of context tiles defined by $C^k_{p,q} = T^k_{p,q} \cap C_A = (T^k_{p,q})^C$.

First, one can prove as in the previous section that sets of triples $T^k_{p,q}$ form the least solution of the set of equations defined, for every p and $q \in Q$, by $T^k_{p,q} = T^0_{p,q}$ as before and, for every $k \geq 0$, by:

$$\begin{aligned} T^{k+1}_{p,q} &= \delta_{p,q} + \sum_{a \in A + \overline{A}} \sum_{(p,r) \in \Delta(a)} \theta(a).T^{k+1}_{r,q} \\ &\quad + \sum_{(p,p') \in \delta(1^+)} \sum_{(r',r) \in \delta(1^-)} C^k_{p',r'}.T^{k+1}_{r,q}(\mathcal{A}) \end{aligned}$$

Indeed, we just mimic in these equation all the possible cases to build a run. Either somme letter $a \in A + \overline{A}$ is red, or a pebble is used. Of course, we check that $T^k_{p,q}$ only depends on languages of the form $T^{k'}_{p',q'}$ with $k' \leq k$, or $C^{k'}_{p',q'} = (T^{k'}_{p,q})^C$ with $k' < k$. That later case shows in particular that no circular dependency involves projections on context tiles. It follows that this system can be solved by induction on $k \in \mathbb{N}$ by Gaussian elimination of variables. Then, for every $k \in \mathbb{N}$, given the k-regular expressions defining languages $T^k_{p,q}$s we conclude by taking $T^k(\mathcal{A}) = \sum_{(p,q) \in I \times F} T^k_{p,q}$. □

Lemma 19 *Every k-regular language of tiles is definable by a finite k-pebble automaton.*

Proof. As for regular languages of tiles, we proceed by induction on the syntactic complexity of k-regular expressions building and combining Multi-pebbles automata. We just detail the construction for the context operators. Given a finite automaton $\mathcal{A} = \langle Q, I, F, \Delta \rangle$ k-recognizing language $T = T^k(\mathcal{A})$ we define automaton $\mathcal{A}' = \langle Q', I', F', \Delta' \rangle$ by taking $Q' = Q \cup \{q_0, q_f\}$ with q_0 and q_f two new states, $I' = \{q_0\}$, $F' = \{q_f\}$ and, for every $a \in A + \overline{A}$, $\Delta'(a) = \Delta(a)$, $\Delta'(1^+) = \Delta(1^+) \cup (\{q_0\} \times I)$ and $\Delta'(1^-) = \Delta(1^-) \cup (F \times \{q_f\})$. One can easily check that automaton $\mathcal{A}'$ recognizes with at most $k+1$ pebbles language $T^C = T^k(\mathcal{A}) \cap C_A$, i.e. $T^{k+1}(\mathcal{A}') = T^C$. The fact k-regular languages only need k-pebbles immediately follows from that construction. □

Corollary 20 *Every language of words recognizable by a k-pebble automata is regular.*

Proof. The proof goes just has for the above proof of Shepherdson theorem. Given a finite k-pebble automata $\mathcal{A}$. The language $L(\mathcal{A}) \subseteq A^*$ of words recognized by $\mathcal{A}$ is defined to be $L(\mathcal{A}) = \{u \in A^* : (1, u, 1) \in T^k(\mathcal{A}) : u_1 = u_3 = 1\}$. Again, given $\# \notin A$, by Lemma 18, language $\#.L(\mathcal{A}).\# = \#.T^k(\mathcal{A}).\# \subseteq \{1\} \times \#A^*\# \times \{1\}$ is regular and we conclude by applying Theorem 9. □

Conclusion

We have shown how McAlister monoid is the adequate domain of interpretation of two-way or many pebble automata on words. As trees can also be embedded in $FIM(A)$, this strongly suggests that some adequate Ree's quotient (by the ideal on non tree-shaped bi-rooted trees) of the free inverse monoid could play the same rôle for tree-walking automata. Such a study is currently under development in connection with the notion of quasi-recognizability defined by the second author [5].

References

[1] Jean-Camille Birget. Concatenation of inputs in a two-way automation. *Theoretical Computer Science*, 63(2):141 – 156, 1989.

[2] Mikolaj Bojanczyk. Tree-walking automata. In *LATA*, volume 5196 of *Lecture Notes in Computer Science*. Springer, 2008.

[3] Noa Globerman and David Harel. Complexity results for two-way and multi-pebble automata and their logics. *Theor. Comput. Sci.*, 169(2):161–184, 1996.

[4] D. Janin. On languages of one-dimensional overlapping tiles. Technical Report RR-1457-12, LaBRI, Université de Bordeaux, 2012.

[5] D. Janin. Quasi-recognizable vs MSO definable languages of one-dimentionnal overlaping tiles. In *37th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2012 (to appear).

[6] Michal Kunc and Alexander Okhotin. Describing periodicity in two-way deterministic finite automata using transformation semigroups. In *Developments in Language Theory*, volume 6795 of *Lecture Notes in Computer Science*, pages 324–336. Springer, 2011.

[7] M. V. Lawson. *Inverse Semigroups : The theory of partial symmetries*. World Scientific, 1998.

[8] Mark V. Lawson. McAlister semigroups. *Journal of Algebra*, 202(1):276 – 294, 1998.

[9] W. D. Munn. Free inverse semigroups. *Proceeedings of the London Mathematical Society*, 29(3):385–404, 1974.

[10] Jean-Pierre Pécuchet. Automates boustrophedon, semi-groupe de birget et monoide inversif libre. *ITA*, 19(1):71–100, 1985.

[11] Jean-Eric Pin. Mathematical foundations of automata theory. Lecture notes, 2011.

[12] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114 –125, april 1959.

[13] J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM J. Res. Dev.*, 3:198–200, April 1959.

[14] Moshe Y. Vardi. A note on the reduction of two-way automata to one-way automata. *Information Processing Letters*, 30:261–264, 1989.